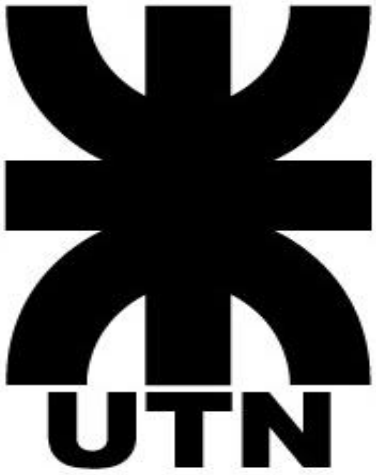

Universidad Tecnológica Nacional
Facultad Regional Concepción del Uruguay

Ingeniería en Sistemas de Información

Sistemas Operativos

Trabajo Práctico Final:
"Ampliación y mejora del shell experimental"

Integrantes del Grupo:

Arrúa, Martin
Lazbal, David
Rivera, Ramiro

Fecha de Entrega:
19/02/2015

Docentes:

Ing. Arellano Gabriel
Ing. Aguiar Osvaldo

```
unit ALR;

INTERFACE

      uses
            baseUnix, Unix, Linux, errors,sysutils,utilidades;

      type
            puntero=^nodo;

            t_procesos= record
                              nombre:string;
                              prioridad:string;
                              numero:longint;
                              directorio:string;
                              estado:string;
                              pid:longint;
                              end;

            nodo= record
                        info:t_procesos;
                        sig:puntero;
                        end;

            tabla=      record
                        cab:puntero;
                        tam:word;
                        indice: word;
                        end;
      var
            alBG: boolean;                      // El programa debe ejecutarse en
bg.
            tablaJobs: Tabla;             // Tabla de tareas activas.
            pidEnEjec: longint;                 // PID del proceso en primer
plano.


procedure agregarArchivo(var arch: text; filename:string; var ok:boolean);
procedure abrirArchivo(var arch: text; filename:string; var ok:boolean);
procedure crearArchivo(var arch: text; filename:string;var ok:boolean);
procedure redirigirSalidaEstandar(var arch:text);
procedure respaldarSalidaEstandar(var respaldo:text);
procedure restaurarStdOut(var arch:text);

{-------------------------------------------------------------------------------
----}

function AnalizarEstado(pid:longint;estado:longint):string;
function procesoFinalizado(estado:longint):boolean;
function codigoFinalizacionProceso(estado:longint):longint;
function procesoSenializado(estado:longint):boolean;
function senialRecibidaPorProceso(estado:longint):longint;
procedure ActualizarEstado(pid:longint;estado:longint);
Procedure SIGTSTP_Recibida(sig : cint);cdecl;
Procedure SIGINT_Recibida(sig : cint);cdecl;
procedure SIGCHLD_Recibida(signal: LongInt; info: psiginfo; context: PSigCon-
text);
procedure instalarManejadores;
```

```
{-------------------------------------------------------------------------
----}

procedure crearTabla;
procedure insertarEnTabla (x:t_procesos);
procedure eliminarDeTabla (pid: longint ;var eliminado:t_procesos);
procedure mostrarTabla;
function damePid (numero:longint):longint;
procedure eliminarPorEstado (var ok:boolean;var correcto:boolean; var elimi-
nado:t_procesos);
procedure limpiarTabla;
function espacio (n:byte):string;
function procesoEnBlanco():t_procesos;
function encontrarProceso (numeroProceso: longint):t_procesos;
function MostrarUno(X:T_procesos):string;


IMPLEMENTATION


Procedure SIGTSTP_Recibida(sig : cint);cdecl;
begin
      if pidenejec<>-1 then
            fpkill(SIGTSTP,pidenejec)
end;


Procedure SIGINT_Recibida(sig : cint);cdecl;
begin
      if pidenejec<>-1 then
            fpkill(SIGINT,pidenejec)
end;


procedure SIGCHLD_Recibida(signal: LongInt; info: psiginfo; context: PSigCon-
text);
{
      Procedimiento a realizar al recibir una sekal SIGCHLD ,
      la cual se recibe cuando el estado de un proceso hijo es modificado.
      Este procedimiento analiza la causa de la modificacion y la refleja en la
Tabla de JOBS
}
begin
      ActualizarEstado
            (
            info^._sifields._sigchld._pid,      //Pid del hijo que envio la se-
nial
            info^._sifields._sigchld._status    // Estado de terminacion para ser
analizado
            );

            {
                  Estructura de psiginfo
                        type psiginfo = ^tsiginfo;

                  Estructura de tsiginfo (en nuestro caso longint = 4)

                        type tsiginfo =
                        record
```

```
                              si_signo: LongInt;            Signal number
                              si_errno: LongInt;            Error code
                              si_code: LongInt;       Extra code (?)
                              _sifields: record       Extra signal information
fields
                                    case LongInt of
                                    0:    (
                                           _pad: array [0..(SI_PAD_SIZE)-1] of
LongInt;          Padding element
                                          );
                                    1:    (
                                          _kill:      record
      Signal number (or status)
                                                      _pid: pid_t;
      Sending process ID
                                                      _uid: uid_t;
      Sending User ID
                                                      end;
                                          );
                                    2:    (
                                          _timer: record                      De-
fault timer
                                                      _timer1: DWord;
      Timer 1 (system time)
                                                      _timer2: DWord;
      Timer 2 (user time)
                                                      end;
                                          );
                                    3:    (
                                          _rt:  record
      Posix compatibility record
                                                      _pid: pid_t;
            Sending process ID
                                                      _uid: uid_t;
            Sending User ID
                                                      _sigval: pointer;
      Signal value
                                                      end;
                                          );
                                    4:    (
                                          _sigchld:   record
      SIGCHLD signal record
                                                            _pid: pid_t;
            Sending process ID
                                                            _uid: uid_t;
            Sending User ID
                                                            _status: LongInt;
      Signal number (or status, SIGCHLD)
                                                            _utime: clock_t;
      User time
                                                            _stime: clock_t;
      System time
                                                            end;
                                          );
                                    5:    (
                                          _sigfault:  record
      SIGILL, SIGFPE, SIGSEGV, SIGBUS record
                                                            _addr: pointer;
            Address (SIGILL, SIGFPE, SIGSEGV, SIGBUS)
```

```
                                                        end;
                                                );
                                        6:      (
                                                _sigpoll:   record
        SIGPOLL record
                                                        _band: LongInt;
                SIGPOLL band
                                                        _fd: LongInt;
                SIGPOLL file descriptor
                                                        end;
                                                );
                                        end;
                        end;
                }
end;


procedure instalarManejadores;
{
        Instala manejadores de señales personalizados para las señales
        SIGCHLD ,SIGTSTP y SIGINT
}
var oldAction,newAction : PSigActionRec; //Punteros hacia registros de acciones
ante seniales
begin
        new(newAction);
        new(oldAction);
        newAction^.sa_Handler:=SigActionHandler(@SIGCHLD_Recibida); //Asigno el
handler a la nueva senial como la funcion SIGCHLD_Recibida
        fillchar(newAction^.Sa_Mask,sizeof(newAction^.sa_mask),#0); //Inicializa-
cion
        newAction^.Sa_Flags:=SA_SIGINFO; {o 4}    //La senial debera devolver in-
formacion detallada, otras acciones son IGNORAR, o DEFAULT
        {$ifdef Linux}              //Compatiblidad Linux
                newAction^.Sa_Restorer:=Nil;
        {$endif}
        if (fpSigAction(SIGCHLD,newAction,oldAction)<>0)then //Chequeo de errores
durante instalacion del nuevo manejador en newAction, el antiguo manejador se
guarda en oldAction
                begin
                writeln('Error en Instalacion: ',fpgeterrno,'.');
                halt(1);
                end;
        fpSignal(SIGTSTP,SignalHandler(@SIGTSTP_Recibida)); //Instalo manejador de
SIGTSTP
        fpSignal(SIGINT,SignalHandler(@SIGINT_Recibida));     //Instalo manejador
de SIGINT
        {
                fpSignal tiene un subconjutno de la funcionalidad de fpSigAction -->
menos funciones
                No necesito tanta informacion sobre el contexto de la recepcion de
TSTP o INT, solo que accion realizar
        }
end;



function procesoFinalizado(estado:longint):boolean;
```

```
begin
      procesoFinalizado:= (wifexited(estado)) ;
end;

function procesoSenializado(estado:longint):boolean;
begin
      procesoSenializado:= (wifsignaled(estado)) ;
end;

function senialRecibidaPorProceso(estado:longint):longint;
begin
      senialRecibidaPorProceso:=wtermsig(estado);
end;

function codigoFinalizacionProceso(estado:longint):longint;
begin
      codigoFinalizacionProceso:=(WExitstatus(estado));
end;


function AnalizarEstado(pid:longint;estado:longint):string;
{
      Analiza el estado devuelto por un fpwaitpid, en funcion
      de la razon de terminacion y/o seniales recibidas por el child
}
begin
if procesoFinalizado(estado) then
                  begin //El programa finalizo por si mismo
                  AnalizarEstado:='Finalizado('+asString(codigoFinalizacionPro-
ceso(estado))+')'; //obtengo su codigo de terminacion (puede haber terminado en
error)
                  end
            else
                  begin
                  if (procesoSenializado(estado)) then
                        begin //El programa envio SIGCHLD por cambio de estado
                        case senialRecibidaPorProceso(estado) of //analizo la
senial que produjo el cambio

                        SIGKILL,
                        SIGTERM,
                        SIGINT:     begin
                                    AnalizarEstado:='Terminado';
                                    fpkill(SIGKILL,pid);
                                    end;

                        SIGCONT:begin
                                    AnalizarEstado:='Corriendo';
                                    end;

                        SIGSTOP,
                        SIGTSTP,
                        SIGTTIN,
                        SIGTTOU:begin
                                    AnalizarEstado:='Detenido';
                                    end;

                        end;
                        end;
```

```
                end;
end;


procedure ActualizarEstado(pid:longint;estado:longint);
var
      Aux:T_procesos;
begin
      eliminarDeTabla(pid,Aux);
      Aux.Estado:=AnalizarEstado(pid,estado);
      insertarEnTabla(Aux);
      dec(tablaJobs.Indice);
end;



{-------------------------------------------------------------------------
----}

procedure agregarArchivo(var arch: text; filename:string; var ok:boolean);
begin
      {$I-}
      assign(arch,filename);
      append(arch);
      ok:=(IOResult=0);
      if not ok then
            crearArchivo(arch,filename,ok);
      {$I+}
end;

procedure abrirArchivo(var arch: text; filename:string; var ok:boolean);
begin
      {$I-}
      assign(arch,filename);
      reset(arch);
      ok:=(IOResult=0)
      {$I+}
end;

procedure crearArchivo(var arch: text; filename:string;var ok:boolean);
begin
      {$I-}
      assign(arch,filename);
      rewrite(arch);
      ok:=(IOResult=0)
      {$I+}
end;

procedure redirigirSalidaEstandar(var arch:text);
{
      el FD Output (Abstraccion de FPC para el FD 1 = STDOUT)
      apunta ahora al ARCH.
}
begin
      fpdup2(arch,output)
end;

procedure respaldarSalidaEstandar(var respaldo:text);
{
```

```
Duplico y devuelvo el FD de Output para luego restaurarlo
}
var
      ok:boolean;
begin
      crearArchivo(respaldo,'salidaEstandarRespaldo',ok);
      if not ok then
            begin
            writeln('Error Critico');
            halt;
            end;
      fpdup2(output,respaldo);
end;

procedure restaurarStdOut(var arch:text);
{
Restaura la STDOUT al FD original, el cual DEBE HABER SIDO PREVIAMENTE RESPAL-
DADO
}
begin
      close(Output);
      assign(Output,'');
      rewrite(output);
      redirigirSalidaEstandar(arch);
      deletefile('salidaEstandarRespaldo');
end;


{-------------------------------------------------------------------------------
----}

function asString(V:longint):string;
{Convierte un número en una cadena}
var S:string;
begin
      STR(V,S);
      asString:=S;
end;

procedure crearTabla;
{Crea e inicializa la tablaJobs}
begin
      tablaJobs.cab:=nil;
      tablaJobs.tam:=0;
      tablaJobs.indice:=0;
end;

procedure insertarEnTabla (x:t_procesos);
{Agregar un elemento proceso a la tablaJobs, ordenando los mismos por número}
var dir,ant,act: puntero;
begin
      new(dir);
      dir^.info:=x;
      if (tablaJobs.cab=nil) or (tablaJobs.cab^.info.numero>x.numero) then
            begin
            dir^.sig:=tablaJobs.cab;
            tablaJobs.cab:=dir;
            end
      else
```

```
            begin
            ant:=tablaJobs.cab;
            act:=tablaJobs.cab^.sig;
            while (act<>nil) and (act^.info.numero < x.numero) do
                  begin
                  ant:=act;
                  act:=act^.sig;
                  end;
            ant^.sig:=dir;
            dir^.sig:=act;
            end;
      inc(tablaJobs.tam);
      inc(tablaJobs.indice);
end;

procedure eliminarDeTabla (pid: longint;var eliminado:t_procesos);
{Elimina un elemento proceso de la tablaJobs, según un pid ingresado}
var ant,act: puntero;
begin
      if tablaJobs.cab<>nil then
      begin
      if tablaJobs.cab^.info.pid = pid then
            begin
                  act:=tablaJobs.cab;
                  tablaJobs.cab:=tablaJobs.cab^.sig;
                  eliminado:=act^.info;
                  dispose(act);
                  dec(tablaJobs.tam);
            end
      else
            begin
            ant:=tablaJobs.cab;
            act:=tablaJobs.cab^.sig;
            while (act<>nil) and (act^.info.pid <> pid) do
                  begin
                  ant:=act;
                  act:=act^.sig;
                  end;
            if act <> nil then
                  begin
                  ant^.sig:=act^.sig;
                  eliminado:=act^.info;
                  dispose(act);
                  dec(tablaJobs.tam);
                  end;
      end;
end;
end;


procedure eliminarPorEstado (var ok:boolean;var correcto:boolean; var elimi-
nado:t_procesos);
{Elimina un elemento proceso de la tablaJobs si su estado actual es "Finalizado"
o "Terminado"}
var ant,act: puntero;
begin
      ok:=false;
      correcto:=false;
      if tablaJobs.cab<>nil then
```

```
      begin
            if (pos('Finalizado',tablaJobs.cab^.info.estado)<>0) or (pos('Termi-
nado',tablaJobs.cab^.info.estado)<>0) then
                  begin
                        eliminado:=tablaJobs.cab^.info;
                        act:=tablaJobs.cab;
                        tablaJobs.cab:=tablaJobs.cab^.sig;
                        dispose(act);
                        dec(tablaJobs.tam);
                        correcto:=true;
                  end
            else
                  begin
                  ant:=tablaJobs.cab;
                  act:=tablaJobs.cab^.sig;
                  while (act<>nil) and ((pos('Finalizado',act^.info.estado)=0)
and (pos('Terminado',act^.info.estado)=0)) do
                        begin
                        ant:=act;
                        act:=act^.sig;
                        end;
                  if act <> nil then
                        begin
                        ant^.sig:=act^.sig;
                        eliminado:=act^.info;
                        dispose(act);
                        dec(tablaJobs.tam);
                        correcto:=true;
                        end
                  else
                        ok:=true;
                  end;
            end
end;


function MostrarUno(X:T_procesos):string;
{Muestra la información de un elemento proceso de la tablaJobs}
begin
MostrarUno:=
      '['+(asString(x.numero))+']'+x.prioridad+espacio(7-length(asString(x.nu-
mero)))+
      x.estado+espacio(16-length(x.estado))+x.nombre+espa-
cio(5)+'('+(asString(x.pid))+')';
end;


procedure mostrarTabla;
{Muestra todos los elementos proceso de la tablaJobs}
var aux:puntero;
begin
      if tablaJobs.cab <> nil then
      begin
            writeln('JobID',espacio (5),'Estado',espacio (10),'Nombre');
            aux:=tablaJobs.cab;
            repeat
                  writeln(MostrarUno(aux^.info));
                  aux:=aux^.sig;
            until aux=nil;
```

```
        end
        else
                writeln('No hay trabajos.');
end;

function damePid (numero:longint):longint;
{Devuelve el pid de un elemento proceso según un número ingresado}
var act:puntero;
begin
        act:=tablaJobs.cab;
        while (act<>nil) and (act^.info.numero<>numero) do
        begin
                act:=act^.sig;
        end;
        if act <> nil then
                damePid:=act^.info.pid
        else
                damePid:=-1;
end;


procedure limpiarTabla;
{Eliminado todos los elementos proceso cuyo estado sea "Finalizado" o "Termi-
nado"}
var ok,correcto: boolean; eliminado: t_procesos;
begin
        ok:=false;
        correcto:=false;
        while not(ok) and (tablaJobs.cab <> nil) do
                begin
                eliminarPorEstado(ok,correcto,eliminado);
                if ((correcto) and (eliminado.pid <> 0)) then
                        writeln(MostrarUno(eliminado));
                end;
end;


function espacio ( n: byte):string;
{Inserta tantos espacios como indique "n"}
var i:byte;
        str:string;
begin
        str:='';
        for i:=1 to n do
                str:=str+' ';
        espacio :=str;
end;


function encontrarProceso(numeroProceso: longint):t_procesos;
{Busca un elemento proceso según un número ingresado}
var aux: puntero; encontrado: boolean; x:t_procesos;
begin
        encontrarProceso:=procesoEnBlanco;
        if tablaJobs.cab <> nil then
        begin
                encontrado:=false;
                aux:=tablaJobs.cab;
                repeat
```

```
                x:=aux^.info;
                if x.numero = numeroProceso then
                      begin
                      encontrado:=true;
                      encontrarProceso:=x;
                      end;
                aux:=aux^.sig;
          until (encontrado) or (aux=nil);
    end;
end;

function procesoEnBlanco():t_procesos;
{"Inicializa" los campos de un elemento proceso}
var x:t_procesos;
begin
    with x do
    begin
          numero:=0;
          nombre:='<< Null process >>';
          pid:=-1;
          estado:='';
          prioridad:='';
          directorio:='';
    end;
    procesoEnBlanco:=x;
end;


end.
```